Trios: A Framework for Interactive 3D Photo Stylization on Mobile Devices

Ulrike Bath
Hasso Plattner Institute
University of Potsdam, Germany
ulrike.bath@student.hpi.uni-potsdam.de

Sumit Shekhar
Hasso Plattner Institute
University of Potsdam, Germany
sumit.shekhar@hpi.uni-potsdam.de

Hendrik Tjabben
Adobe Systems Engineering GmbH
Hamburg, Germany
tjabben@adobe.com

Amir Semmo
Digital Masterpieces GmbH
Potsdam, Germany
amir.semmo@digitalmasterpieces.com

Jürgen Döllner
Hasso Plattner Institute
University of Potsdam, Germany
juergen.doellner@hpi.uni-potsdam.de

Matthias Trapp
Hasso Plattner Institute
University of Potsdam, Germany
matthias.trapp@hpi.uni-potsdam.de

***Abstract*—For decades, Image-based Artistic Rendering (IB-AR) has been successfully employed to simulate the appeal of traditional artistic styles for enhanced visual communication. Recently, 3D photography has emerged as a new medium that provides an immersive dimension compared to 2D photos. The possibility to change the viewpoint, depicting parallax effects is mesmerizing. We present *Trios*, an interactive mobile app that combines the vividness of IB-AR with immersive 3D photos. *Trios* implements an end-to-end pipeline for the acquisition of data, generation of a 3D photo in the form of a Layered Depth Image (LDI), and its artistic rendering. The app allows the user to either capture the input data or load existing data from the device. As part of the generation step, users can set the number of layers used for representation of 3D photos. Finally, with different artistic filters and their parameterization users can stylize either an individual semantic layer or all layers simultaneously. The complete pipeline runs at interactive frame rates and the final output is obtained as a compact video, which can easily be shared. Thus, it serves as a unique interactive tool for digital artists interested in creating immersive artistic content.**

***Index Terms*—3D photos, image stylization, mobile devices**

I. Introduction

A. Motivation

Traditional 2D photo captures a scene as a frozen moment in time. Recently 3D photos have emerged as a new medium to make such moments more immersive [1]. We refer to 3D photo as a representation which introduces scene parallax – difference in the apparent position of scene-objects due to change in viewpoint (and not the traditional stereo-pair images for perceived 3D effect). The ability to explore parallax effects, especially on the flat-screen of a mobile device, is compelling [2]. On the other hand, with the advancement in mobile graphics advanced stylistic rendering techniques have been successfully deployed on mobile devices [3], [4]. In this work, we aim to extend the visual-richness of image-stylization approaches for 3D photos which are deployable on a mobile-device. As compared to traditional stylization techniques, 3D photos offer new possibilities with respect to stylization and visualization. Moreover, a mobile-based approach will have implications in terms of ease of use.

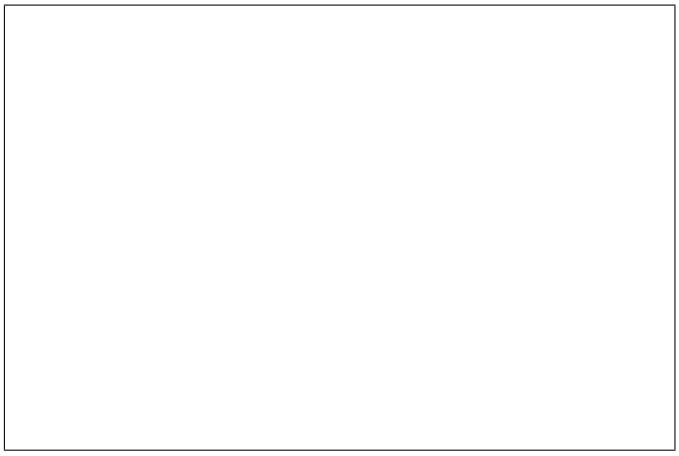


Fig. 1: *Trios* is a mobile app that enables users to render and stylize 3D photos. The user interface provides interactive control over a variety of artistic filters, both classical and neural, as well as rendering aspects of 3D photos.

B. Problem Statement & Challenges

To this end, we develop a mobile-based framework for generation and stylization of 3D photos given RGB or RGB-D input data (Fig. 1). We identify and address the following challenges:

a) Interactivity: Most of the existing approaches for creating 3D photos do not involve users in the generation step. The end user only views the final output while the generation pipeline remains a black-box. We offer a simple user-interface to interact with the generation and the stylization aspects.

b) Consistency: For traditional 2D photos any editing should be spatially consistent across semantic similar regions for visually aesthetic output. In case of 3D photos, the above requirement becomes paramount as any spatial inconsistencies also reflect temporally while viewing/exporting the 3D photo. We address the above via (semantic-)segment-wise stylization of the image and smooth virtual-camera movement while viewing/exporting the 3D photo.

c) Throughput: To achieve consistent output while maintaining interactivity, we require fast throughput on a mobile device. We achieve the above by making use of GPU-aligned data structures and Apple's proprietary graphical processing (Metal) Application Programming Interfaces (APIs).

C. Approach & Contributions

To address these challenges, our proposed framework adopts the following approach. As the first step, if no depth data is given, we estimate the depth for the given input image using a mobile version of a state-of-the-art depth-estimation technique [5]. The given/estimated depth data is used to decompose the RGB image into layers at different depth levels and is represented as a LDI [6], stored in a 2D texture array for Graphics Processing Unit (GPU)-based processing. Unlike Kopf *et al*. [2] we do not convert the LDI representation into a 3D-mesh, as LDI-space allows for efficient image-based stylization. For each layer, potential dis-occlusions which might get visible due to viewpoint change are identified and inpainted. The inpainted layers can be visualized as a 3D photo within the app via viewpoint variations induced due to device movement or via traversing a particular trajectory by the virtual camera. Subsequently, the generated 3D photo can be exported as a video file. For spatially consistent stylization, the input image is divided into (semantic-)segments and the user performs stylization on a per-segment basis. Similar to input-image, the stylized image is decomposed into depth-based layers which can then be exported as a 3D photo.

Our contributions are summarized as follows, we propose (1) a novel framework for 3D photo generation and stylization on mobile devices, and (2) a respective user-interface for interactive editing of stylization parameters and camera animations.

II. Background & Related Work

A. 3D Photo Generation

Hedman *et al*. [1] introduce the concept of 3D photography by using a set of input photos to construct a 3D panorama. In a follow-up work, Hedman and Kopf [7] propose a novel optimization that speeds up the panorama generation step by two orders of magnitude. The focus of both of these work is to capture large panoramas to be viewed in a Virtual Reality (VR) setup. The approaches are inconvenient regarding usability and requires capturing multiple photos. Mildenhall *et al*. [8] propose an algorithm that guides users to capture a grid of sampled views, wherein each sample is expanded to a local light-field which are then fused for virtual scene exploration. Similar to previous techniques, the above requires capturing of multiple images for light-field fusion. Shih *et al*. [9] propose the first method for converting a single RGB-D image into a 3D photo employing a multi-layer representation for novel view synthesis. The authors make use of LDI as a multi-layer representation for efficient editing [6], [10]. We also employ this representation for novel view synthesis and exploration. Jampani *et al*. [11] utilize depth-aware inpainting for improved segmentation and layering, thereby preserving fine image details in the foreground. Kopf *et al*. [2] for the first time proposed a method wherein the entire 3D photo generation pipeline is carried out on a mobile device. Our framework is also implemented on a mobile-device, however, unlike previous approach we provide user control in the generation process. Further, we allow interactive stylization of 3D photos for enhanced visual aesthetics.

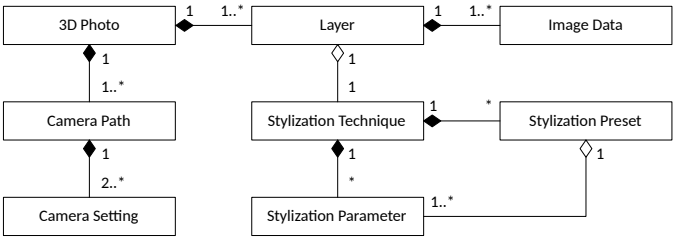


Fig. 2: Data model for 3D photo stylization. A 3D photo basically resembles an Layered Depth Image (LDI) comprising multiple color and depth layers, each referencing a stylization technique that is parameterized by a number of parameters, which are grouped to presets. For rendering and export, a number of virtual camera animations can be used.

B. Image Stylization on Mobile Devices

Due to advances in mobile graphics hardware, on-device image stylization is not only becoming feasible but also increasingly popular via casual creativity apps. Durschmid *et al*. [12] present a generic GPU-based app that allows to design on-device stylization components by reusing building blocks. Pasewaldt *et al*. [4] showcase a broad range of IB-AR techniques running on a mobile device. The above mobile-based methods consider only RGB data as input. Recently, Shekhar *et al*. [13] demonstrate depth-based stylization methods running interactively on a mobile device. As part of our approach we employ depth data only for 3D photo generation and as future work would also incorporate it for stylization. Note that there are already consumer mobile applications (e.g., Loopsie, PopPic, Parallax, DazzCam) that allow on-device 3D photo generation and limited editing. However, unlike these, we provide a broad range of advanced IB-AR techniques for editing the 3D photo. Further, we provide more control to the user in terms of generation and editing.

III. Data Model for 3D Photo Stylization

Prior to focusing on 3D photo synthesis and stylization (Sec. IV), this section briefly describes the basic data model used by our approach. Fig. 2 shows an overview of the respective data structures.

3D Photo: A 3D photo is represented as an LDI [6], i.e., a number of layers, as well as respective camera animation data required for viewing and export.

Layer: Fundamentally, a layer associates required (color and depth) and optional (normal, mask) image data with reference to a depth-level.

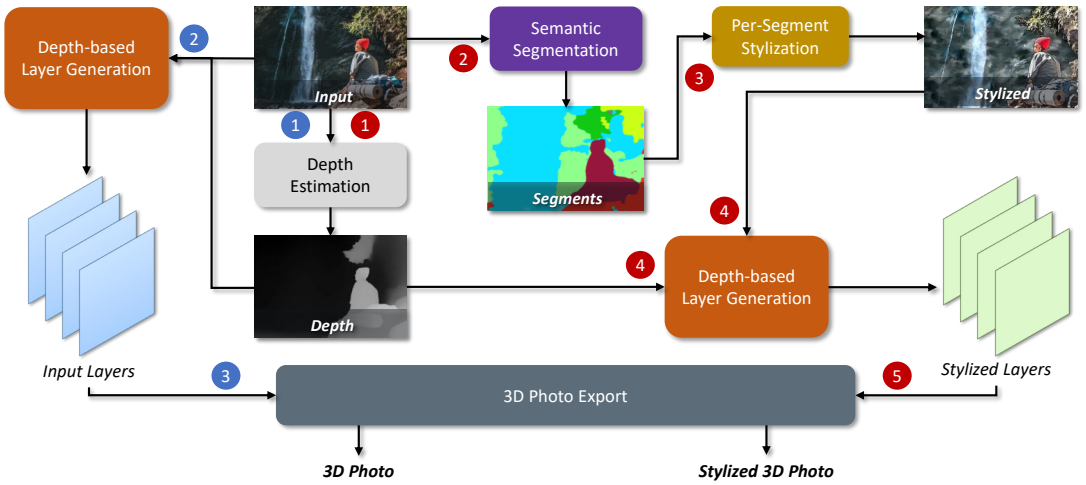


Fig. 3: Schematic overview of the processing pipeline implemented in our app *Trios*, where steps 1-3 depict the creation of a *3D Photo* and another set of steps 1-5 showcase the creation of a *Stylized 3D Photo*.

Image Data: In addition to color and depth images, image data instances can comprise masks and intermediate representations required for processing.

Stylization Technique: An instance of a stylization technique defines the order of algorithms applied to a semantic segment yielding an abstraction or stylized image.

Stylization Parameter & Preset: Stylization techniques are parameterized by a number of parameters whose values can be controlled directly by a user or defined via presets.

Camera Path & Setting: To represent animations of the virtual camera, for 3D photo synthesis, a camera path instance stores an ordered list of camera settings. A camera setting comprises a 2D camera position, 2D look-to vector, and a zoom parameter.

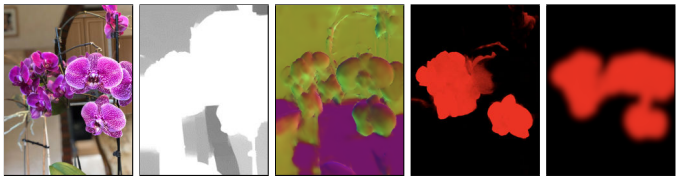
IV. Method for 3D Photo Stylization

Fig. 3 shows an overview of the presented framework. Its modular design comprises the following conceptual processing stages, which are described in the remainder of this section.

Input Data Acquisition & Preprocessing: This stage acquires the data required to synthesize and stylize a 3D photo. It can consume RGB and RGB-D data and optionally compute additional raster data used within the framework and perform any preprocessing if required (Sec. IV-A).

LDI Generation & Inpainting: For 3D photo synthesis, our approach is based on the concept of LDIs. For it, this stage separates individual colors layers and perform inpainting necessary to generate a plausible parallax effect (Sec. IV-B).

Per-Segment Stylization: The depth-based layer generation do not respect image semantics. To address this issue and produce a spatially consistent output, we divide the image into semantic-segments. This core stage enables the above and further application of stylization techniques on a per-segment basis (Sec. IV-C).



(a) Input (b) Depth (c) Normal (d) Matte (e) Salience

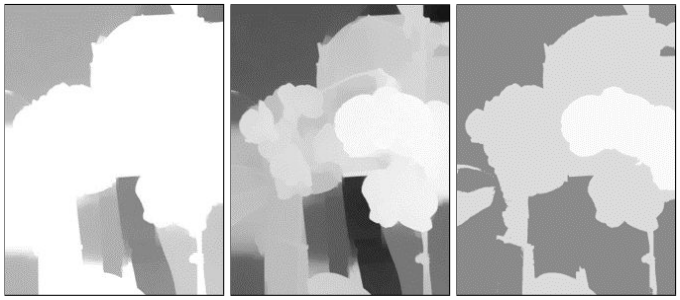
Fig. 4: Examples of additional raster data that can be automatically derived based on a color input image and used as input for image stylization and 3D photo rendering.

Rendering & Export: This stage exports the synthesized 3D photo animation as a video based on the virtual camera path specified by the user (Sec. IV-D).

A. Acquisition and Preprocessing of Input Data

This stage prepares the input data to be used in the subsequent stages of our framework.

a) Input Data Acquisition and Generation: Fig. 4 shows examples of data our framework can consume for 3D photo stylization. Besides depth, it computes normal vectors for surface orientation [14], as well as matte and saliency data; the last two acquired using the Apple Vision framework. To support a potentially wide range of mobile devices, this stage can handle two types of input data: (1) RGB-only images and (2) RGB-D images, depending on the respective mobile hardware available to a user. In case of RGB-only input image, the pre-processing stage uses MiDaS [5] to compute relational depth information based on color values. In case the depth data is provided by the device depth-sensors, it is usually of lower spatial resolution than the respective color image. For it, the depth map is upsampled to the color image resolution using a standard upsampling filter, e.g., joint-bilateral upsampling [15].



(a) Acquired depth D (b) Norm. depth D_N (c) Quantized depth D_Q

Fig. 5: Visualization of exemplary depth data during different stages of pre-processing. First, the acquired depth values (a) are normalized to span the complete range of possible depth values (b). Subsequently, normalized values are then quantized uniformly into a number of bins specified by the user (c).

(a) Input (b) Background (c) Midground (d) Foreground

Fig. 6: Example visualization of separating an input image (a) into individual LDI layers (b)-(d), based on quantized depth data (Fig. 5c). White indicates pixels associated to other layers.

b) Preprocessing of Depth Data: Following to that, further depth data processing is performed as depicted in Fig. 5. After acquisition (Fig. 5a), the depth map (D) is normalized (Fig. 5b) based on a pre-computed depth histogram. The histogram yields the minimum ($d_{\min}$) and maximum ($d_{\max}$) depth values. Normalization is then performed:

$$D_N(x,y) = \frac{D(x,y) - d_{\min}}{d_{\max} - d_{\min}}$$

Subsequently, the normalized values are thresholded (Fig. 5c) based on uniform quantization:

$$D_Q(x,y) := \begin{cases} \lfloor D_N(x,y) \cdot b \rfloor & D_N(x,y) < 1 \\ b-1 & \text{otherwise} \end{cases}$$

The number of bins $b \in \mathbb{Z}^{0+}$ represents layers of unique depth value, and can be set by the user as a parameter. In general, the number of depth layers account for the resulting rendering quality and processing performance, i.e., lower enables faster rendering, higher increases visual quality – depending on the distribution of depth values. In our experiments, we found that $b = 3$, i.e., the separation between background, midground, and foreground is sufficient for most scenes (Fig. 6).

(a) Input color (b) Coverage (c) Inpainting

Fig. 7: Visualization of the layer processing stages performed by the framework prior to the stylization and rendering of 3D photos. For the input color layer (a), regions are inpainted that would dis-occlude during camera animation (c). All other pixel data will remain unchanged, visualized by the coverage (b) (white: requires inpainting, gray: no inpainting required).

B. LDI Computation and Inpainting

Based on the pre-processed depth data, this stage first performs LDI generation by separating the RGB-D data into individual layers of unique depth complexity. Following to that, for each layer the RGB regions that possibly become subject to disocclusions during 3D photo rendering are inpainted. These steps yield a number of RGB-D layers that can be stylized individually in an art-directed way (Sec. IV-C).

a) Layer Separation: Fig. 6 shows an example of the separated LDI layer, based on the pre-processing results depicted in Fig. 5c. The layer segmentation is performed based on a per-pixel level using depth data as follows. For each depth bin $i \in 0, \ldots, b-1$, the input image I is copied into a respective layer L_i. The transparency value of pixels is set to zero if the depth value at that location does not belong to the respective bin.

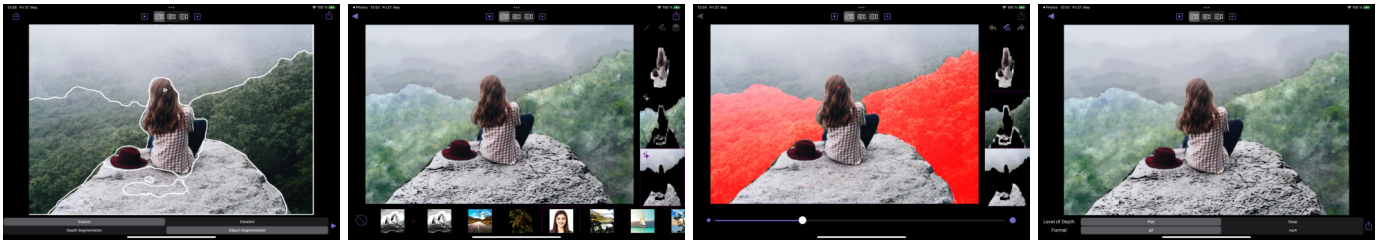
b) Inpainting: Subsequent to the layer separation, inpainting is performed for each layer L_i with $i = 0, \ldots, b-2$, i.e., the foreground layer remains unaffected. Inpainting is required to hallucinate color information that was occluded during acquisition and becomes visible during 3D photo rendering. For each layer L_i, a coverage value c at each pixel position (x, y) is determined based on the depth values in D_Q:

$$c(x,y) := \begin{cases} 1 & D_Q(x,y) > i \\ 0 & \text{otherwise} \end{cases}$$

With the coverage data c (c.f. Fig. 7b), we make use of Bilateral Filter for inpainting similar to Shekhar *et al.* [13]. The filter is applied on a per-layer basis for the image regions that are qualified with $c(x,y) = 1$ and within a specified distance to the visible pixels. For bilateral filter parameters, we use $r = 4$, $\sigma_s = 5$ for spatial as well as $\sigma_r = 12$ for the range. The above is an efficient approach which gives visually plausible output. However, as part of future work it can be improved using learning-based techniques.

(a) Original image (b) Depth map (c) Semantic segmentation map (d) Per-Layer Stylization. (e) Per-Segment Stylization

Fig. 8: For an input image (a) depth-map (b) is used to break image into multiple layers. For a spatially consistent stylization we perform semantic segmentation (c). The stylization can be performed using a per depth-based layer approach (d) or a per semantic-segment approach (e). Note how semantic segmentation reduces spatial inconsistencies due to stylization.



(a) Segmentation Screen (b) Stylization Screen (c) Touch-Up Screen (d) Export Screen

Fig. 9: Overview of the main user interface screens provided by our prototypical application for a stylization session. Starting with the segmentation screen (a), a user can load or capture an image and define the basic properties of the 3D photo, e.g., the segmentation method and desired number of segment-layers. The stylization screen enables the user to select and adjust stylization effects (b). If a user is not satisfied with layer borders, it can be adjusted by directly drawing on the image (c). After finishing the editing process, the export screen provides result preview and allow the user to export different file formats (d).

C. Per-Segment Stylization

The layers generated based on depth do not respect the image semantics, (Fig. 8). To obtain a consistent output, semantically similar regions should be stylized in a similar fashion. Thus, we divide the input image into segments based on a semantic segmentation model. We integrate different artistic rendering effects for stylizing these segments [16]. Specific to *Trios*, these comprise variants of Cartoon [17], Watercolor [18], Oil paint [19], and Hatching [20]. In case, integrated stylization techniques require depth data, it is facilitated via our framework. Sec. VI-A describes the possibilities of the per-layer stylization approach.

D. Rendering for Preview and Export

This stage synthesizes a 3D photo animation for preview and exports based on the (stylized) LDI layer and camera settings. For rendering a single frame of 3D photo animation, we implement the approach given by Shade *et al.* [6]. To achieve interactive performance, we make use of GPU-aligned implementation based on custom data structures for image storage and representation. Our framework allows setting the number of LDI layers generated during export to achieve varying intensities of parallax.

V. User Interface

We prototypically implement the proposed framework based on iOS and iPadOS. The code is based on Swift, UIKit, CoreImage, CoreML, and Metal APIs. However, the implementation methodology is not device-specific and can also be extended for other high-end mobile devices. Our prototypical app reflects our method's structure by offering a dedicated screen for each step. The modular structure allows for easy transitions between these (Fig. 10). Further, the user experience is designed to accommodate editing on multiple levels-of-control [21].

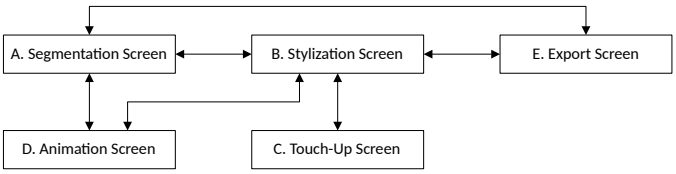


Fig. 10: Diagram showing the control flow between the individual screens of our prototypical application.

Fig. 9 shows an overview of the main views, which functionalities are briefly outlined in the remainder of this section. As a main feature for visual feedback during editing, every screen allows for selecting and playing a 3D photo animation. The animation is described via a virtual camera path wherein a user can select from a number of pre-defined paths.

A. Segmentation Screen

Fig. 9a shows an example of the segmentation screen, the first screen provided after on-boarding. It allows the user to

load a RGB(-D) image or acquire one using built-in camera functionality. The screen allows for previewing the synthesized 3D photo using pre-defined or custom animation modes (Sec. V-D). Further, the user can choose between a "coarse" ($b = 3$) or "detailed" ($b = 5$) LDI representation. In addition to depth-based layers (IV-C) a semantic-segmentation-map is also created here. To provide visual feedback to the user, the boundaries of the respective layers are depicted using white lines. This separation is only used for the stylization phase.

At this point, a user can already choose to export the 3D photo animation (Sec. V-E) which represents the standard functionality of existing 3D photo apps.

B. Stylization Screen

For design reasons, we assume that the target audience is familiar with raster-image editing apps and therefore decide to re-use Graphical User Interface (GUI) concepts from common image-editing applications [22]. It offers different levels-of-control, ranging from choosing stylization presets (high-level) to adjusting individual parameters (low-level) [21]. An icon indicates, if a layer has a stylization technique applied.

For it, the stylization screen (Fig. 9b) presents the individual layers on the right screen side using an ordered list, descending from background (top) to foreground (bottom). Upon layer selection, the user can choose from various artistic effect presets displayed below the preview image. Thus, a user can rapidly switch between stylization variants and control the overall results in an art-directed manner. Additionally, the stylization screen offers access to further low-level layer-management operations, such as parameter control (Fig. 11a), touch-up for the layer mask (Fig. 11b, Sec. V-C), as well as merging selected layers. Specific parameter controls can be used to make adjustments to the selected preset. The touch-up button opens a screen described in the next paragraph.

(a) (b) (c)

Fig. 11: Layer controls.

C. Touch-Up Screen

Fig. 9c shows the screen that offers layer-based touch-up functionality. If the user is not satisfied with the generated segments, the borders can be corrected by simply drawing on the image. The respective pixels are added to the mask of the selected segment. The current segment is highlighted with a red overlay. The drawn path is displayed directly and added to the path after the touch is finished. The radius of the brush can be adjusted with the slider at the bottom.

D. Camera Controls & Edit Screen

In order to enable easy exploration of preliminary editing results, our GUI offers control over the virtual camera in every screen, located above the 3D photo preview. It enables a user to play a camera animation and select from three different predefined animation modes as follows. The pre-selected camera animation mode (Fig. 12b) interpolates between a fixed number of virtual camera positions and orientations.

(a) (b) (c) (d) (e)

Fig. 12: Camera controls.

Further, for fast exploration of the result and detection of possible artifacts due to discontinuities or disoccluded areas, the GUI enables the traversal along a spiral path (Fig. 12c) – usual for 3D photo exploration. Finally, the position and orientation of the virtual camera can be directly controlled by the device gyroscope or accelerometer data (Fig. 12c). To allow for additional control over the virtual camera, a user can specify custom camera animations by switching to the camera edit screen (Fig. 12e). Here, the user can specify their own camera animation path by replacing the predefined one. For it, camera settings are generated by storing positions tapped on the image view. Subsequently, the camera settings can be manipulated with respect to the look-to vector, zoom level, traversal timing, and interpolation functions.

E. 3D Photo Export Screen

Finally, the user can export the 3D photo stylization results. For this, *Trios* offers a dedicated screen (Fig. 9d) for settings regarding the "level-of-depth" as well as the output file "format". The number of layers reflects the perceived intensity of the resulting parallax effect. We choose $b = 3$ for a "flat" and $b = 7$ for the "deep" option. This way, the user does not need to edit all layer that contribute to the parallax effect during final 3D photo rendering and can focus on the major composition elements (e.g., background, foreground, etc.). With respect to the export-file format, our prototype currently supports videos and as future work we would also like to include animated images.

VI. Results & Discussion

This section evaluates our approach regarding runtime performance (Sec. VI-B) and discusses limitations (Sec. VI-D) by means of different application examples (Sec. VI-A).

A. Application Examples

Fig. 13 shows exemplary results generated using our framework and a prototypical mobile application. In average, the users required $1\,\mathrm{min}$ to $3\,\mathrm{min}$ for stylization. The per-segment stylization approach offers a high degree of flexibility. For example, all segments can be stylized with the same stylization technique but using a single or multiple different presets. Usually, users tends to apply more aggressive stylization on background segments and maintain high detail in the foreground. Further, users are allowed to use different stylization techniques per-segment or do not even apply any stylization to a particular segment.

B. Performance Evaluation

a) System & Setup: We test the performance of *Trios* using the following setup. Tests on mobile were executed using an iPad Pro 3$^{\mathrm{rd}}$ generation equipped with an Apple A12X Bionic and $4\,\mathrm{GB}$ Random Access Memory (RAM). With respect to the test data, we perform runtime analysis using

(a) *Cartoon stylization* with more abstraction towards the background.

(b) *Watercolor stylization* using different accent colors per layer.

(c) *Pencil-hatching stylization* applied on each layer except the main focus.

Fig. 13: Images stylized by *Trios* using different presets and stylization parameter configurations on a per-layer basis.

TABLE I: Runtime performance of the individual stages in our framework different input image resolutions.

Input Image Resolution	Pre-processing	#Layers	Segmentation	Stylization per layer	Rendering	Overall
HD (1280 × 720 pixels)	0.15 s	3	0.14 s	0.4 s (0.15 s × 3)	0.11 s	0.85 s
		5	0.31 s	0.75 s (0.15 s × 5)	0.15 s	1.21 s
FHD (1920 × 1080 pixels)	0.17 s	3	0.15 s	0.57 s (0.19 s × 3)	0.16 s	1.05 s
		5	0.32 s	0.95 s (0.19 s × 5)	0.24 s	1.68 s
QHD (2560 × 1440 pixels)	0.19 s	3	0.15 s	0.63 s (0.21 s × 3)	0.17 s	1.14 s
		5	0.33 s	1.05 s (0.21 s × 5)	0.27 s	1.84 s

images of three different resolutions: High Definition (HD) (1280×720 pixels), Full High Definition (FHD) (1920×1080 pixels), and Quad High Definition (QHD) (2560×1440 pixels).

b) Run-time Performance Results: Tab. I shows the runtime performance results for each pipeline stage with increasing image resolutions. We record the processing time for the steps of segmentation, stylization of all layers, and the final rendering. One can observe, that the runtime performance of each step scales with the image resolution and the number of layers to stylize. The type of effect, selected for stylization, has negligible impact on the overall performance. During the export, the rendered 3D photo is displayed on the screen and is simultaneously written to the memory. Thus, saving the 3D photo takes approximately as long as the final result visualization. Note, that for depth estimation and/or object segmentation we use trained neural-network models. These models are only loaded once and have an initial loading overhead of approx. $5\,\mathrm{s}$.

c) Memory Consumption: The prototypical app itself has a storage size of $1.8\,\mathrm{GB}$ on the iPad. The memory consumption of our prototype scales linearly with the resolution of the used image. For an image of spatial resolution of 1920×1080 pixels, the memory usage is approx. $35\,\mathrm{MB}$ without stylization and approx. $135\,\mathrm{MB}$ with stylization applied. The final 3D rendering step increases the memory usages to $275\,\mathrm{MB}$. The exported 3D file itself, e.g., M4V, of $5\,\mathrm{s}$ has a size of $7\,\mathrm{MB}$. Thus, the application has a reasonable memory footprint.

C. Usability Evaluation

The prototype was presented at an international conference on computer graphics using the same setup described in Sec. VI-B. We gave a brief introduction to *Trios* to approximately 40 people, who then choose example images or took photos and edited these accordingly. A user spend on average $2\,\mathrm{min}$ to $5\,\mathrm{min}$ to create a stylized 3D photo. Most users were familiar with the general concepts of 3D photos and stylization, thus immediately understood the concept of layer-wise combination of both. The working modes were well understood, however some functionalities had to be pointed out repeatedly, e.g., multi-selection and layer merging.

The device motion was mostly used to view the parallax effect since it had the most immersive effect for the users. When handed the device, users often directly tried to move the photos using device rotation. However, the responding transformation of the 3D photo was often found to be slightly contra-intuitive or not always reliable.

Of particular interest to most users was switching between different stylization presets rather than using the fine-tuning option. Generally, mostly two different stylizations were chosen – background and foreground. Regarding the type of stylization, either strongly varying styles were chosen for more contrast or similar stylization techniques with different level-of-abstraction to increase depth sensation. Overall the user's feedback was positive. Especially people from non-technical background were excited to have on-device 3D stylization. However, as per the feedback, the experience can be further improved by more reliable device motion and better layer separation.

D. Discussion and Future Work

Our goal is to develop a framework for interactive stylization of 3D photos on mobile devices. To this end, we deploy depth-estimation and semantic-segmentation neural-network models on-device. We observe that the optimized mobile-based models perform significantly worse than their desktop counterparts,

while still giving plausible results for our purpose. For high-quality results, better depth-quality and inpainting techniques are required. However, increased layer numbers and a sophisticated inpainting algorithm impacts interactivity.

Although, our app shows the feasibility of our framework and achieve sufficient interactive characteristics, we plan to address several aspects as future work. The presented modular framework provides the basis for straightforward integration of alternative or additional image processing operations. For example, we plan to use live-photos or videos to improve the resulting inpainting quality, e.g., using the work of [9]. With a combination of depth- and semantic-estimation, the resulting depth map can be further improved to avoid objects being split to different layers [23]. Further, depth-map upsampling can be improved using guided filtering [24] and can form the basis to implement stylized atmospheric effects [13].

VII. Conclusions

In this work, we present a framework for implementing 3D photo stylization techniques on mobile devices. Our approach is based on layered depth-images and proposes a modular concept for data acquisition, pre-processing, stylization, and rendering of 3D photos. We demonstrate and evaluate the feasibility by providing an initial implementation based on Apple consumer devices. Our integrated approaches enable users to rapidly create stylized variants of 3D photos, which can easily be shared using common interchange file formats such as animated images or videos.

Acknowledgments

This work was partially funded by the German Federal Ministry of Education and Research (BMBF) through grants 01IS18092 (“mdViPro”) and 01IS19006 (“KI-LAB-ITSE”) and the Research School on “Service-Oriented Systems Engineering” of the Hasso Plattner Institute.

References

[1] P. Hedman, S. Alsisan, R. Szeliski, and J. Kopf, “Casual 3d photography,” *ACM Trans. Graph.*, vol. 36, no. 6, nov 2017. [Online]. Available: https://doi.org/10.1145/3130800.3130828

[2] J. Kopf, K. Matzen, S. Alsisan, O. Quigley, F. Ge, Y. Chong, J. Patterson, J.-M. Frahm, S. Wu, M. Yu, P. Zhang, Z. He, P. Vajda, A. Saraf, and M. Cohen, “One shot 3d photography,” *ACM Trans. Graph.*, vol. 39, no. 4, jul 2020. [Online]. Available: https://doi.org/10.1145/3386569.3392420

[3] J. E. Kyprianidis, J. Collomosse, T. Wang, and T. Isenberg, “State of the ”art”: A taxonomy of artistic stylization techniques for images and video,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 5, pp. 866–885, 2013.

[4] S. Pasewaldt, A. Semmo, J. Döllner, and F. Schlegel, “Becasso: Artistic image processing and editing on mobile devices,” in *SIGGRAPH ASIA 2016 Mobile Graphics and Interactive Applications*, ser. SA ’16, 2016.

[5] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, “Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020.

[6] J. Shade, S. Gortler, L.-w. He, and R. Szeliski, “Layered depth images,” in *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH ’98. New York, NY, USA: Association for Computing Machinery, 1998, p. 231–242. [Online]. Available: https://doi.org/10.1145/280814.280882

[7] P. Hedman and J. Kopf, “Instant 3d photography,” *ACM Trans. Graph.*, vol. 37, no. 4, jul 2018. [Online]. Available: https://doi.org/10.1145/3197517.3201384

[8] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar, “Local light field fusion: Practical view synthesis with prescriptive sampling guidelines,” *ACM Trans. Graph.*, vol. 38, no. 4, jul 2019. [Online]. Available: https://doi.org/10.1145/3306346.3322980

[9] M.-L. Shih, S.-Y. Su, J. Kopf, and J.-B. Huang, “3d photography using context-aware layered depth inpainting,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8028–8038.

[10] M. Trapp and J. Döllner, “Efficient Representation of Layered Depth Images for Real-time Volumetric Tests,” in *Theory and Practice of Computer Graphics*, I. S. Lim and W. Tang, Eds. The Eurographics Association, 2008.

[11] V. Jampani, H. Chang, K. Sargent, A. Kar, R. Tucker, M. Krainin, D. Kaeser, W. T. Freeman, D. Salesin, B. Curless, and C. Liu, “Slide: Single image 3d photography with soft layering and depth-aware inpainting,” in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 12 498–12 507.

[12] T. Dürschmid, M. Söchting, A. Semmo, M. Trapp, and J. Döllner, “Prosumerfx: Mobile design of image stylization components,” in *SIGGRAPH Asia 2017 Mobile Graphics & Interactive Applications*, ser. SA ’17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3132787.3139208

[13] S. Shekhar, M. Reimann, M. Mayer, A. Semmo, S. Pasewaldt, J. Döllner, and M. Trapp, “Interactive Photo Editing on Smartphones via Intrinsic Decomposition,” *Computer Graphics Forum*, 2021.

[14] A. R. Zamir, A. Sax, N. Cheerla, R. Suri, Z. Cao, J. Malik, and L. J. Guibas, “Robust learning through cross-task consistency,” in *CVPR*. Computer Vision Foundation / IEEE, 2020, pp. 11 194–11 203.

[15] J. Kopf, M. F. Cohen, D. Lischinski, and M. Uyttendaele, “Joint bilateral upsampling,” *ACM Trans. Graph.*, vol. 26, no. 3, p. 96–es, jul 2007. [Online]. Available: https://doi.org/10.1145/1276377.1276497

[16] A. Semmo, M. Trapp, J. Döllner, and M. Klingbeil, “Pictory: Combining neural style transfer and image filtering,” in *ACM SIGGRAPH 2017 Appy Hour*, ser. SIGGRAPH ’17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3098900.3098906

[17] H. Winnemöller, S. C. Olsen, and B. Gooch, “Real-time video abstraction,” *ACM Trans. Graph.*, vol. 25, no. 3, p. 1221–1226, jul 2006. [Online]. Available: https://doi.org/10.1145/1141911.1142018

[18] A. Bousseau, M. Kaplan, J. Thollot, and F. X. Sillion, “Interactive watercolor rendering with temporal coherence and abstraction,” in *Proceedings of the 4th International Symposium on Non-Photorealistic Animation and Rendering*, ser. NPAR ’06. New York, NY, USA: Association for Computing Machinery, 2006, p. 141–149. [Online]. Available: https://doi.org/10.1145/1124728.1124751

[19] A. Semmo, J. Döllner, and F. Schlegel, “Becasso: Image stylization by interactive oil paint filtering on mobile devices,” in *Proceedings SIGGRAPH Appy Hour*. New York: ACM, 7 2016, pp. 6:1–6:1.

[20] A. Semmo and S. Pasewaldt, “Graphite: Interactive photo-to-drawing stylization on mobile devices,” in *ACM SIGGRAPH 2020 Appy Hour*, ser. SIGGRAPH ’20. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3388529.3407306

[21] T. Isenberg, “Interactive NPAR: What Type of Tools Should We Create?” in *Proc. NPAR*, ser. Expressive ’16. Goslar, DEU: Eurographics Association, 2016, p. 89–96.

[22] M. Klingbeil, S. Pasewaldt, A. Semmo, and J. Döllner, “Challenges in user experience design of image filtering apps,” in *SIGGRAPH Asia 2017 Mobile Graphics & Interactive Applications, Bangkok, Thailand, November 27 - 30, 2017*, M. Billinghurst and W. Rungjiratananon, Eds. ACM, 2017, pp. 22:1–22:6. [Online]. Available: https://doi.org/10.1145/3132787.3132803

[23] S. Niklaus, L. Mai, J. Yang, and F. Liu, “3d ken burns effect from a single image,” *ACM Trans. Graph.*, vol. 38, no. 6, nov 2019. [Online]. Available: https://doi.org/10.1145/3355089.3356528

[24] K.-L. Hua, K.-H. Lo, and Y.-C. F. Frank Wang, “Extended guided filtering for depth map upsampling,” *IEEE MultiMedia*, vol. 23, no. 2, pp. 72–83, 2016.